CLAIMS:

1. A method of translating an object-oriented computer program comprising:

(a) translating the program bytecode into machine independent virtual processor code which uses an instruction set of a virtual processor; and

(b) translating the virtual processor code into native code which uses an instruction set of a physical processor.

2. A method as claimed in claim 1 in which the program bytecode includes a class file, the class file being converted into one or more virtual processor tools which use the instruction set of the virtual processor.

3. A method as claimed in claim 2 in which the class file includes a plurality of methods, and which some or all the methods in the class file are converted to a respective virtual processor tool.

4. A method as claimed in claim 2 in which the class file includes a call to a method, and in which the virtual processor code provides a call to a corresponding tool.

5. A method as claimed in claim 2 in which the class file includes a reference to a field, and in which the virtual processor code provides a fixup tool for use in locating the field.

6. A method as claimed in claim 5 in which the fixup tool is arranged to

return a constant fixup value which is representative of the offset of the said field within an object.

7. A method as claimed in claim 6 including linking the virtual processor code and determining the constant fixup value in dependence upon virtual processor code which has been translated from another class file.

8. A method as claimed in claim 6 or claim 7 in which the fixup tool returns a value which is used to patch a method which gets or puts the value of a field.

9. A method as claimed in claim 2 in which the virtual processor code has, included within it at a plurality of points, fixup instructions which indicate that the code at the said points has to be modified by the respective fixup instruction prior to use.

10. A method as claimed in claim 7 in which the fixup instructions provide instructions as to how the native code can reference another class, or a field or method in another class.

11. A method as claimed in claim 9 or claim 10 in which the fixup instructions are transferred, substantially functionally unaltered, by the native translator into the native code; the fixup instructions being replaced with native instructions when the native code is bound on the said real physical processor.

12. A method as claimed in any one of claims 1 to 11 in which the bytecode is stack-based, and in which the virtual processor code is register-based.

13. A method of executing an object oriented computer program comprising translating the program into native code as claimed in any one of the preceding claims, and executing the native code on the physical processor.

14. A method as claimed in claim 13 when dependent upon claim 2 including binding the translated tools into a task, and executing the task in native code on the physical processor.

15. A computer system adapted to carry out a method as claimed in any one of the preceding claims.

16. A method as claimed in any one of claims 1 to 12 which further includes:

(c) translating the virtual processor code into a different native code which uses an instruction set of a second physical processor.

17. A method as claimed in claim 13 or claim 14 including executing the different native code on the second physical processor.

18. A computer system adapted to carry out a method as claimed in claim 16 or claim 17.

19. A distributed computer system comprising a server including a store for storing virtual processor code, said code being a machine-independent representation of an object oriented computer program, and a plurality of remote client devices in communication with the server, each client device including a client processor, a native translator arranged to translate the virtual processor code into native code which uses the instruction set of the respective

client processor, and a native code store; the system including transmission means for transmitting the virtual processor code from the server to the client devices.

20. A distributed computer system as claimed in claim 19 in which the transmission means consists of or includes a wireless network.

21. A distributed computer system as claimed in claim 20 in which the client devices are mobile phones.

22. A distributed computer system as claimed in claim 20 in which the client devices are hand-held computers.

23. A distributed computer system as claimed in claim 19 or claim 20 in which the client devices are hand-held games consoles.

24. A distributed computer system as claimed in claim 19 in which at least one of the client devices includes a first type of client processor and in which at least another of the client devices includes a second type of client processor, using a different instruction set from that of the first type.

25. A distributed computer system as claimed in any one of claims 19 to 24 in which the server is further arranged to translate the object-oriented computer program from bytecode into virtual processor code.

26. A method as claimed in any one of claims 2 to 11, or claim 12 when dependent upon claim 2, including verifying the integrity of the class bytecode,

and of any external calls.

27. A method as claimed in any one of claims 2 to 11, or claim 12 when dependent upon claim 2, in which the class file is a Java class file.

28. A method as claimed in any one of claims 1 to 12, 26 or 27 in which the step of translating the program bytecode into virtual processor code is carried out by a first translator program which is itself written in virtual processor code.

29. A method as claimed in any one claims 1 to 12, 26, 27 or 28, in which the step of translating the virtual processor code into native code is carried out by a second translator program which is itself written in virtual processor code.

30. A computer program for executing a method as claimed in any one of claims 1 to 12 or 26 to 29.

31. A data carrier which carries a computer program for executing a method as claimed in any one of claims 1 to 12 or 26 to 29.

32. A data stream representative of a computer program for executing a method as claimed in any one of claims 1 to 12 or 26 to 29.